UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/784,026	02/20/2004	Laurent D. Hasson	YOR920030638US1	1688

34663 7590 12/10/2007

MICHAEL J. BUCHENHORNER
8540 S.W. 83 STREET
MIAMI, FL 33143

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

NOTIFICATION DATE	DELIVERY MODE
12/10/2007	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

MICHAEL@BUCHENHORNER.COM
ANA@BUCHENHORNER.COM

Office Action Summary	Application No.	Applicant(s)	
	10/784,026	HASSON ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *20 February 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-14* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-14* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *20 February 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed 2/20/2004.

Claims 1-14 have been submitted for examination.

Specification

2. The disclosure is objected to because of the following informalities: the term 'sever' in line 24 pg. 4 (para 0015) is a typographical error and appropriate correction is required.

Claim Objections

3. Claim 11 is objected to because of the following informalities: the phrase recited as 'computing the signature of the object *check* the Map ... if the object' appears to contain grammatical impropriety because the action *check* appears not clearly in accord with any subject. Appropriate correction is required.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claim 13 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

The Federal Circuit has recently applied the practical application test in determining whether the claimed subject matter is statutory under 35 U.S.C. § 101. The practical application test requires that a " useful, concrete, and tangible result" be accomplished. An "abstract idea" when practically applied is eligible for a patent. As a consequence, an invention, which is eligible for patenting under 35 U.S.C. § 101, is in the "useful arts" when it is a machine, manufacture, process or composition of matter, which produces a concrete, tangible, and useful result. The test for practical application is thus to determine whether the claimed invention produces a "useful, concrete and tangible result".

The current focus of the Patent Office in regard to statutory inventions under 35 U.S.C. § 101 for method claims and claims that recite a judicial exception (software) is that the claimed

invention recite a practical application. Practical application can be provided by a physical transformation or a useful, concrete and tangible result. The following link on the World Wide Web is for the United States Patent And Trademark Office (USPTO) policy on 35 U.S.C. §101. <http://www.uspto.gov/web/offices/pac/dapp/opla/preognotice/guidelines101_20051026.pdf>

Specifically, claim 13 discloses a system comprising a *web server* (i.e. Specifications: element 108 in Fig. 1) *comprising Java objects*, a program for converting with instructions for *identifying, determining, introspecting, creating*, and *generating*. The system as claimed and understood based on the *web server--* being viewed as Java objects, amounts to a mere listing of software instructions or objects. According to the § 101 Guidelines (see Annex IV(a), pg 53-54) for it falls under the subject matter identified as 'Functional Descriptive Material' and cannot be considered sufficiently supported with hardware in order to realize the software into real-world tangible results. The claim as a whole is rejected for leading to a non-statutory subject matter.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Jim Conallen , "Modeling Web Application Architectures with UML", October 1999, ACM, Vol 42, No 10, pp. 63-70 (hereinafter Conallen), in view of Chung et al, "Modeling Web Applications Using Java and UML Related Technologies', Proceedings of the 36th HICSS'03, IEEE 2002 pp. 1-10 (hereinafter Chung).

As per claim 1, Conallen discloses a server method for converting objects of a first type into objects of a second type, the method comprising:

identifying one or more object classes of the first type (e.g. *Web pages ... class abstraction of logical behavior* – pg. 68, bottom L col. to top R col.; Fig. 3, pg. 68) ;

determining instance data based on the classes of the first type (e.g. Fig. 3; UML ... custom icon – pg. 68, bottom L - Note: runtime nature of Web pages being passed from client to server being reconstructed at server page reads on OO instance data being introspected at server to map a UML construct, i.e. such instance leading to a UML icon -- see *variable number of instances* – L top col., pg. 70);

introspecting each class of the first type; automatically creating an artifact representing a software model (e.g. pg. 68: bottom L col. to top R col. - Note: runtime data of Web pages being from client to server reads on derived *artifacts* -- details inside each UML iconic structures -- based on OO instances from Web pages introspected at server to derive a corresponding UML construct; Fig. 3); and

generating one or more converters (e.g. <form> - R col, bottom, pg. 69; HTML input tags - pg. 70, L top para), each converter being based on the artifact and is configured for receiving the instance data.

Conallen does not explicitly disclose generating JavaScript code for recreating the instance from the classes as objects of the second type, for display on a browser. However, the client/server system by Conallen clearly discloses Javascript based client side communicating with server UML-based framework to create Web applications (e.g. "Java Script item" - Fig. 4, pg. 69) and Web forms or frame navigation supporting client pages (pg. 70, L top paras;

navigation of Web pages - pg. 64, L col; Fig. 5, pg. 69;) to be utilized in relation with server in business transactions and a converter to output formatted stream to address requests the client browser navigation scheme or underlying HTML or Java script execution (see HTML formatted stream ... sent back - pg. 66, middle R col.; Fig. 4, pg. 69). The need to provide Javascript-based form of HTML/Web executable constructs to support the client for effectuating a business transaction or to render, at the Jvscript-based client side the data sent back from the server is suggested from the above. The type of components that can be modeled in UML (see Chung: Fig. 6, pg. 9) and deployed at workstations as scripts (see Chung: sec 4, 5.1, pg. 3) are taught in Chung; and accordingly, Chung discloses client-side script programming being Javascript (see pg. 3, bottom L col) and Javascript to validate data inside a Web HTML frame. In view of the need to visualize the content of runtime transaction-related pages as see in Conallen(see Fig. 5, pg. 69) and the client side role played by Javascript as by Chung, it would have been obvious for one skill in the art at the time the invention was made to implement the conversion in Conallen's server developing tool so that the returned formatted stream as by Conallen be effectuated to contain the server-side generated Javascript constructs (i.e. OO class of second type). One would be motivaged to do so because the generated JVscript code would support the deployment of business web-based applications navigation as in Conallen's client side, the support as conveyed in Chung's establishing UML/Javascript relationship along with necessary Javascript dependency within deployment of application content (e.g. Email application, see Chung Fig. 6) at the requesting client as endeavored in Conallen's paradigm.

As per claim 2, Conallen discloses wherein the objects of the first type are Java-based objects (see Web page ... Javascript – pg. 65, R col.; Fig. 4, pg. 69) and (in view of Chung) the objects of the second class are JavaScript-based objects.

As per claim 3, Conallen discloses wherein the converters are invoked at runtime to generate JavaScript code (in light of the combination with Chung) required to recreate instance data from the JavaBeans into objects in the browser; determining, for each property in a class, whether the property is an attribute (see class attribute – pg. 68, bottom L col.). But Conallen does not disclose that applets in a client Web Application (see Fig. 2, pg. 68) comes from server building based on class objects from Javabeans. But Chung teaches JavaBeans for server-side applications and modeling a EmailBean via introspecting the class attributes (see bottom L col. pg. 3 to top R col. pg. 4; *attributes. JavaBean* – sec 5.3, pg. 5 R col). Based on the above, it would have been obvious for one skill in the art at the time the invention was made to implement the client side applets by Conallen from forming the code in the server UML framework based on beans as taught above for which the analysis of class attributes as in Chung's reusable package (e.g. *what packages are used ... easily by using class notation in UML* - sec 5.3, pg. 4) prove (i) an efficient way of reusing Java code and (ii) easy Java Class mapping with UML notation, thereby enabling proper support of the client applet executed at Conallen's browser runtime, as mentioned above.

As per claim 5, Conallen does not explicitly disclose outputting a JavaScript code for the value of the attribute. But based on the attribute of Javascript (see JavaScript Participant, Fig. 2-3,pg. 68; JavaScript item, Fig. 4, pg. 69) from Conallen, and the rationale in claim 1, this

outputting in Jvscript code including value of the attribute would also have been obvious in view of the above obvious reasons.

As per claim 6, Conallen does not explicitly disclose creating a buffer to hold an exported object value or values if the property is an array. At the time the invention was made, it was well-known concept that *Javascript* does not support a predefined type as with strongly typed languages like C++ or Java, and that an array in Javascript can have no fixed bound as it by itself grows dynamically upon execution unless its bounds have to be customized by user's code; and this is suggested in Conallen (see pg 70: type-less language). Based on the communication paradigm to provide stream of formatted data in order to support the client-side rendering of Web pages via server-side generating of Javascript code as rationalized in claim 1 and Conallen's analysis of bean-class attributes as set forth above, it would have been obvious for one skill in the art at the time the invention was made so that Conallen's exporting of attributes for a Javascript code to the client Web pages for rendering would include provision for Conallen's attributes enumerated or listed in an array (see Fig. 2-3, pg. 68), in view of the above dynamic expanding of array size; that is, provision in form of creating buffer to hold values of the exported array because this would accommodate to the type bound deficiency of Javascript with the runtime requirements of a given business transaction effectuated using Javascript on the client pages.

As per claim 7, the limitation creating an array to hold object IDs for the referenced objects for each object in the array for holding all the attributes of Java class implementing a transaction functionality in Conallen in form of Javascript code would falls under the obviousness rationale as set forth in claims 1 and 6.

As per claim 8, Conallen does not explicitly disclose computing a signature of the object if the object is in a map. But the analysis of UML constructs entail a plurality of map instance between a UML notation representing a client functionality and a corresponding server side UML equivalent (see Fig. 2-3, pg. 68); and since the providing of attributed in Java being effectuated to support export of object class to implement the Javascript code for the client applications, the class being viewed in Conallen's mapping would necessitate a constructor for the Javabeans as set forth in claim 3. Hence the limitation as creating a signature for a beans (or its constructor) would have been an obvious limitation based on the code generating of Java constructs as set forth in claim 3.

As per claims 9-10, Conallen is not explicitly disclosing the steps of calling an object; generating an export ID; outputting the JavaScript code to declare the object called as the converter for the object; and adding the ID to the array. But these are obvious steps for standard construction of a class beans in view of claim 3, the array of claim 7, and the plurality of attributes identified (see claims 6-7) when creating the buffer and array for exporting the Javascript-based object (using a export ID object map as in claim 8) including populating the array with attributes ID for such export object destined for the client Web page execution environment as set forth in claim 1; that is Conallen discloses (in view of the above obvious constructor signature and array ID) outputting a JavaScript array to hold the exported IDs.

As per claim 11, the limitation as to check whether the object already in the Map get its export ID is deemed a obvious step for validating a Java construct to enable all the attributes to be included in the Javascript implemented with a Map for export to the client Javascript-based runtime, lest the execution of said Javascript would incur for example, null pointer exception.

As per claim 12, Conallen (in view of claim 9) discloses calling the converter for the object based on the rationale in claims 1 and 3; and outputting the buffer for the exported object values based on the rationale of claim 6.

As per claim 13, Conallen discloses an information processing system comprising:

a web server comprising one or more Java objects (Fig. 3, pg. 68); and

a program for converting the Java Objects to JavaScript objects (Fig. 2-3, 4, pg. 68-69); wherein the program comprises instructions for:

a) identifying one or more object classes of a first type;

b) determining instance data based on the classes of the first type;

c) introspecting each class of the first type;

d) automatically creating an artifact representing a software model; and

e) generating one or more converters, each converter being based on the artifact and is configured for receiving the instance data.

Conallen does not explicitly disclose generating JavaScript code for recreating the instance from the classes as objects of a second type, for display on a browser. But this limitation has been addressed in claim 1.

As per claim 14, Conallen (in view of Chung) disclose a computer readable medium comprising program instructions for:

a) identifying one or more object classes of a first type; b) determining instance data based on the classes of the first type; c) introspecting each class of the first type; d) automatically creating an artifact representing a software model; and e) generating one or more converters, each converter being based on the artifact and is configured for receiving the instance data and

generating (refer to the combination with Chung in claim 1) JavaScript code for recreating the instance from the classes as objects of a second type, for display on a browser.

Conclusion

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Tuan A Vu
Patent Examiner,
Art Unit 2193
December 05, 2007